IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2018, No. 1, pp. 74–100. DOI:10.13154/tosc.v2018.i1.74-100

Searching for Subspace Trails and Truncated Differentials

Gregor Leander[1], Cihangir Tezcan[1,2] and Friedrich Wiemer[1]

[1] Horst Görtz Institute for IT-Security, Ruhr-Universität Bochum, Bochum, Germany
{gregor.leander,friedrich.wiemer}@rub.de
[2] Informatics Institute, Department of Cyber Security, CYDES Laboratory, and Department of Mathematics, Middle East Technical University, Ankara, Turkey
cihangir@metu.edu.tr

Abstract. Grassi et al. [GRR16] introduced subspace trail cryptanalysis as a generalization of invariant subspaces and used it to give the first five round distinguisher for Aes. While it is a generic method, up to now it was only applied to the Aes and Prince. One problem for a broad adoption of the attack is a missing generic analysis algorithm.
In this work we provide efficient and generic algorithms that allow to compute the provably best subspace trails for any substitution permutation cipher.

Keywords: Subspace Trail Cryptanalysis · Truncated Differentials · Tools

1 Introduction

Despite good progress in the last decades, especially within the Aes competition and more recently within the area of lightweight cryptography, some fundamental questions of the design and analysis of block ciphers (or hash-functions or cryptographic permutations) still remain open. Several of those fundamental questions can be found in the area of differential cryptanalysis and its variants. In differential cryptanalysis we are interested in studying the behavior of the output differences of (parts of) the cipher for a given fixed input difference. In order to analyze the distribution of those differences, we (i) are required to make simplifying assumptions (like assuming independent round keys), (ii) argue about average behavior only, and (iii) study differential characteristics instead of differentials. While the general experience is that for most practical applications, the simplified analysis is a sufficiently good heuristic, the situation is clearly unsatisfactory from a scientific perspective. Any progress here would significantly improve our understanding of block ciphers.

Another example of a related area for which a strict analysis without simplifying heuristics is still missing is the topic of truncated differentials. Informally, for truncated differentials, instead of trying to understand the exact behavior of the output difference, one restricts to understand certain patterns that appear in the differences with an unusual high probability. Truncated differentials have been introduced by Knudsen [Knu95] more than 20 years ago and since then been used in the analysis of many symmetric primitives, e. g., [BN14; Gra17; KRW99; Tez16]. Surprisingly, even the case of truncated differentials with probability one is not fully understood yet, as this work shows.

Recently, new interest in truncated differentials has been triggered by [GRR16] where the notion of subspace trail cryptanalysis has been defined and used to derive new interesting properties of the Aes. As subspace trails are closely related to truncated differentials

Received: 2017-09-01, Revised: 2017-11-23, Accepted: 2018-01-23, Published: 2018-03-01

with probability one, those recent works again highlight that it is important to finally completely understand this topic.

A one round subspace trail can be captured as follows: Given a function F on n bit strings, a subspace trail is specified by two subspaces $U, V \subseteq \mathbb{F}_2^n$, such that any coset of U is mapped to a coset of V, that is

$$\forall a \in \mathbb{F}_2^n \quad \exists b \in \mathbb{F}_2^n \text{ such that } F(U + a) \subseteq V + b$$

and we denote this by $U \to V$.

Extending this concept to r rounds of an iterated block cipher with round function F, we consider subspace trails given by a tuple of vector spaces $(U_0, \ldots, U_r)$ such that for all i we have

$$U_i \to U_{i+1}.$$

The important question, both for the design as well as for the analysis of block ciphers, is how to identify the most powerful subspace trails, where most powerful basically corresponds to covering as many rounds as possible. In the case of truncated differentials, several heuristics have been used to solve this problem so far. For SPN ciphers, where each round function consists of a layer of parallel S-boxes followed by a linear mapping, the two most common ones are to ignore the details of the S-box and to restrict to the cases where U_0 only activates one S-box. While intuitively this approach seems to cover the best subspace trails, it seems hard to exclude the existence of better subspace trails outside those special cases. Indeed, to underline that the heuristic of activating a single S-box is not sufficient in general we provide several examples (see Section 3) where the best subspace trails actually activate more than one S-box.

For the designers of block ciphers it would be very convenient to exclude the existence of subspace trails without making any restrictions and thus to avoid having to base the security arguments of a cipher upon heuristics.

For attackers it is interesting to see if attacks could be improved by avoiding those restrictions. As an example, the subspace trail used in [GRR17] does not make use of any specific properties of the AES S-box. Thus, one important question raised is, if those results on AES could actually be improved by taking the specific structure of the AES S-box into account.

1.1 Our Contribution

In this work we rigorously analyse subspace trails for SPN ciphers. As a result of our considerations, we provide efficient and generic algorithms that can be applied to any SPN cipher and compute the longest subspace trails without any heuristics or restrictions.

As a first step in Section 2, after fixing our notation and recalling basic facts, we recall that it is actually possible to efficiently compute the entire subspace trail for any number of rounds efficiently, *given the starting subspace* U_0. Thus, the task of finding the best solution actually boils down to choosing the best starting spaces U_0. However, apriori there is a huge choice of possible starting spaces and it is clearly inefficient to simply try all possible U_0. We thus have to exhibit a way to reduce the choice of U_0 to a suitable number in such a way that we are *guaranteed* to not exclude the best choices. Our consideration here show an interesting difference depending on whether the S-box used in the cipher has linear structures or not.[1] Makarim and Tezcan [MT14] already observed an influence of linear structures on truncated differentials, but restricted themselves to linear structures in the coordinate functions of S-boxes. By considering all linear structures of an S-box, we are able to fully understand its influence on subspace trails.

[1] We like to note that both situations, that is S-boxes with and without linear structures occur in actual cipher designs.

More precisely, if the S-box used in the cipher does not have any linear structures (see Section 4), we can prove that the approach sketched above, that is to ignore the details of the S-box and to only consider the case where U_0 activates a single S-box, always results in the strongest subspace trail. More technically, we show that in this case for any subspace trail the subspaces U_i are without loss of generality direct products of subspaces that are aligned with the S-box layer. Note that the AES S-box does not have any linear structures. In particular, this shows that an attacker cannot hope to improve the work of [GRR17] by taking the details of the AES S-box into account.

In the case when the S-box actually does have linear structures (see Section 5), the situation is slightly more complicated and the choice for U_0 remains huge. However, we show that simply switching from the input subspace of the first S-box layer to the output subspace of the first S-box layer results in a simple and efficient workaround. Here, we can prove that it is sufficient to consider a very limited choice of at most $k2^n$ choices for the output subspace of the first S-box layer, where k is the number of parallel S-boxes and n is the input size of a single S-box. The price to pay for this switch is that it is in general unclear if any of those trails can actually be extended backwards through the first S-box layer. However, using this approach we are able to provably bound the longest subspace trail. More precisely, our bound is either tight (in the case where we can actually extend trails backwards) or off by at most one round (in the case where we cannot).

All our algorithms are efficient for any concrete instance of a block cipher we are aware of. We run the algorithms on a number of ciphers and report on the results in the respective parts of Sections 4 and 5. Note that while our results will most likely not lead to new attacks on the ciphers we have been investigating, the main point is that we can now provably exclude the existence of such attacks.

We implemented all algorithms in C and SAGE; the SAGE the source code is listed in the Appendix A.

1.2 Related Work

Besides the subspace trail on AES [GRR17], subspace trail cryptanalysis has been applied to PRINCE in [GR16]. A paper that is technically related, but focuses on invariant subspaces instead of subspace trails is [LR17].

Grassi et al. [GRR16] noted the strong connection between subspace trails and truncated differentials with probability one. Actually subspace trails are a special case of the latter. Truncated differentials are commonly used for impossible differential cryptanalysis, developed by Knudsen [Knu98] and Biham et al. [BBS99a; BBS99b]. Indeed, truncated differentials with proability one, and thus subspace trails as a special case, can be used to construct impossible differentials, e. g. by looking for two trails that miss in the middle.

Several automatic tools were proposed for finding impossible differentials. However, none of them is able to *provably* find the best impossible differential efficiently.

The majority of these do not consider S-box details [Cui+17; Kim+03; Luo+09; Luo+14; Sun+16; WW12]. Only few attempts were done to understand the influence of the S-box layer. Wang and Jin [WJ17] improve on [Sun+16] by analysing this influence in the case of the AES. An attempt to partially cover the S-box influence is the notion of undisturbed bits, developed in the context of probability one truncated differentials [Tez14]. Makarim and Tezcan [MT14] started to describe undisturbed bits with linear structures of coordinate functions of S-boxes. Exploiting these bits results in the best known impossible differentials for some block ciphers [TTD14; Tez14; Tez16].

Derbez and Fouque [DF16] and Sasaki and Todo [ST17] tackle the influence of an S-box without restricting to a special one. The first develop a generic algorithm working on a system of equations which describes the algorithm under scrutiny. While this in principle allows to handle a very large class of block ciphers, it comes at the cost of an increased runtime. To solve this problematic long runtime, the authors revert to handling (parts of)

the S-box as a black box. Moreover, the truncated differentials considered are restricted to the case where state bits may be active or passive, while more general subspaces are not handled. The second use a different approach, namely mixed integer linear programming (MILP). Due to size constraints, the MILP model is not able to handle 8-bit S-boxes and also block sizes of 256 or 512 bits seem to be out of reach with this technique. Additionally, the authors did not consider all possible starting differences, but focused on activating only single S-boxes.

2 Notations and Preliminaries

We start by recalling some definitions and state some useful lemmata, which we need later.

2.1 Basics

By $\mathbb{F}_2$ we denote the finite field with two elements and by $\mathbb{F}_2^n$ the n dimensional vector space over $\mathbb{F}_2$. For the canonical inner product in $\mathbb{F}_2^n$ we write $\langle \cdot, \cdot \rangle$, that is for $x, y \in \mathbb{F}_2^n$ we have $\langle x, y \rangle = \sum_i x_i y_i$. Note that most of the following is included to facilitate a self containing work.

Definition 1 (Orthogonal Complement)**.** Let $V \subseteq \mathbb{F}_2^n$ be a subspace. Then

$$V^\perp \coloneqq \{a \in \mathbb{F}_2^n \mid \forall v \in V : \langle a, v \rangle = 0\}$$

is the *orthogonal complement* (or perpendicular complement) of V.

We are going to use two well known properties of the orthogonal complement later. Namely, that the complement of a direct sum is the direct sum of its complements, i. e.

$$(V_1 \times \cdots \times V_n)^\perp = V_1^\perp \times \cdots \times V_n^\perp,$$

and given two subspaces U, V it holds that

$$U \subseteq V \Rightarrow V^\perp \subseteq U^\perp.$$

One of the two most common cryptanalytic techniques is differential cryptanalysis. Here properties of the derivative of a vectorial Boolean function are studied.

Definition 2 (Derivative)**.** Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$. The *derivative of F in direction α* is defined as

$$\Delta : \mathbb{F}_2^n \times (\mathbb{F}_2^n \to \mathbb{F}_2^m) \to (\mathbb{F}_2^n \to \mathbb{F}_2^m)$$
$$\Delta_\alpha(F)(x) \coloneqq F(x) + F(x + \alpha).$$

The second notation we utilize in this work are linear structures. First, recall the definition of a linear structure for a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$, see e. g. [CV05; Eve88; Lai95]:

Definition 3 (Linear Structures in Boolean functions)**.** Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$. The set of *linear structures of f* is

$$\mathrm{LS}(f) = \{u \mid \Delta_u(f)(x) = c \quad \forall x \in \mathbb{F}_2^n, c \in \mathbb{F}_2\}.$$

We call $u \in \mathrm{LS}(f)$ for which the derivative is constant zero (or one) a *0-linear structure* (or *1-linear structure*).

Note that $\mathrm{LS}(f)$ and $\mathrm{LS}^0(f)$ are subspaces and $\mathrm{LS}(f)$ is partitioned by the above distinction into $\mathrm{LS}(f) = \mathrm{LS}^0(f) \cup \mathrm{LS}^1(f)$, where $\mathrm{LS}^1(f)$ is either empty or a coset of $\mathrm{LS}^0(f)$.

This definition can be naturally extended to vectorial Boolean functions.

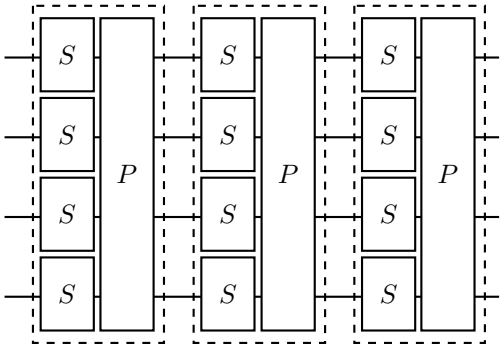

Figure 1: Three rounds of a generic SPN with four S-boxes.

Definition 4 (Linear Structures in vectorial Boolean functions)**.** Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$. The set of *linear structures of* F is

$$\mathrm{LS}(F) \coloneqq \{(\alpha, u) \in \mathbb{F}_2^m \times \mathbb{F}_2^n \mid \langle \alpha, \Delta_u(F)(x) \rangle = c \quad \forall x \in \mathbb{F}_2^n, c \in \mathbb{F}_2\} \tag{1}$$

We define the set L_u as

$$\mathrm{L}_u(F) \coloneqq \{\alpha \in \mathbb{F}_2^n \mid (\alpha, u) \in \mathrm{LS}(F)\},$$

Just like $\mathrm{LS}(f)$, we can partition $\mathrm{L}_u(F) = \mathrm{L}_u^0(F) \cup \mathrm{L}_u^1(F)$ by fixing the constant c in Eq. (1) to either 0 or 1, and where the later is either empty or a coset of the first. Finally, like with $\mathrm{LS}(f)$, linear structures with constant $c = 0$ are called *0-linear structures*, with $c = 1$ they are called *1-linear structures*.

In this work we deal with substitution permutation ciphers, see Fig. 1. That is, we consider iterated ciphers where each round consists of an S-box layer followed by a linear mapping. As it does not affect any of our considerations, we ignore the key-addition and the key-scheduling.

Turning to the parts of an SPN construction, the most common design of an S-box layer is to use the same S-box in parallel for the whole state, which is covered in the next definition. We stick to this sort of S-box layer for the remainder of the work. Extension to the more general case of different S-boxes is straightforward.

Definition 5 (S-box layer)**.** Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be an S-box. Then an *S-box layer* F^k is the parallel application of F for k times:

$$F^k : (\mathbb{F}_2^n)^k \to (\mathbb{F}_2^n)^k$$
$$F^k(x_1, \ldots, x_k) \coloneqq (F(x_1), \ldots, F(x_k))$$

And finally, we call an S-box *active*, if it has a non-zero input/output difference, otherwise *passive*.

2.2 Subspace Trails

Grassi et al. [GRR16] introduced subspace trail cryptanalysis as a generalization of invariant subspaces. We recall the definition of subspace trails.

Definition 6 (Subspace Trail)**.** Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$. Linear subspaces $U, V \subseteq \mathbb{F}_2^n$ are called a *(one round) subspace trail*, if

$$\forall a : \exists b : F(U + a) \subseteq V + b$$

We denote this by $U \xrightarrow{F} V$. We write $U \overset{F}{\nrightarrow} V$, if they do not form a subspace trail, i. e.

$$\exists a : \forall b : F(U + a) \nsubseteq V + b$$

An $r + 1$-tuple of subspaces $(U_1, \ldots, U_{r+1})$ is called a *subspace trail* (over r rounds), if

$$U_i \xrightarrow{F} U_{i+1} \quad \forall i \in \{1, \ldots, r + 1\}.$$

We can identify some trivial subspace trails:

- $U = \{0\}$, $V = \{0\}$
- pick any $U \subseteq \mathbb{F}_2^n$, $V = \mathbb{F}_2^n$

Besides these, if $U \xrightarrow{F} V$, then for all $U' \subseteq U$ and $V' \supseteq V$ we have $U' \xrightarrow{F} V'$. The intuition here is that decreasing U will never result in a bigger V and increasing V does of course also not change the possible output differences in the trail. This leads to the following definition.

Definition 7 (Essential Subspace Trail)**.** Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ and $U \subseteq \mathbb{F}_2^n$, $V \subseteq \mathbb{F}_2^m$. If $U \xrightarrow{F} V$ forms a subspace trail, i. e. $F(U + a) \subseteq V + b$, and if for all subspaces U' and V' of $\mathbb{F}_2^n$ the two properties (2) and (3) hold, we call $U \xrightarrow{F} V$ an *essential subspace trail*:

$$\forall U' \supset U : U' \overset{F}{\nrightarrow} V \tag{2}$$

$$\forall V' \subset V : U \overset{F}{\nrightarrow} V' \tag{3}$$

The two properties above ensure that we cannot increase U, nor decrease V without destroying the subspace trail property. Essential subspace trails are clearly the most interesting ones from an attacker perspective. Another important observation is the following.

Corollary 1. *Let $U_1 \xrightarrow{F} U_2$ be a subspace trail through F and $V_1 \subseteq U_1$ a subspace contained in U_1. Then there exists a subspace $V_2 \subseteq U_2$, s. t. $V_1 \xrightarrow{F} V_2$ is also a subspace trail:*

$$\begin{array}{ccc} U_1 & \xrightarrow{F} & U_2 \\ \cup\mathsf{I} & & \cup\mathsf{I} \\ V_1 & \xrightarrow{F} & V_2 \end{array} \tag{4}$$

In other words, it is actually enough to consider one dimensional starting subspaces only, when trying to identify the longest possible subspace trail. That is, the effect of reducing the initial dimension of the starting subspace can only cause longer subspace trails, not shorter ones. Thus when we are using this to bound the subspace trail lengths we are potentially only overestimating the length. However, as even in this case the number of starting spaces to consider grows exponentially with the block size, this is still clearly unfeasible for most common block sizes.

We next elaborate on the relation between subspace trails and truncated differentials with probability one.

2.3 Truncated Differentials

Truncated differentials were introduced in [Knu95]. The notation was later generalized to subspaces of differences in [BLN15; BLN16]. Following this line, we define truncated differentials as affine spaces of differences.

Definition 8 (Truncated Differential)**.** Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$. A *truncated differential* of probability one is defined by a pair of affine subspaces $(U \subseteq \mathbb{F}_2^n, s \in \mathbb{F}_2^n)$, and $(V \subseteq \mathbb{F}_2^m, t \in \mathbb{F}_2^m)$, for which

$$\forall \alpha \in U : \forall x \in \mathbb{F}_2^n : F(x) + F(x + \alpha + s) \in V + t.$$

We write $U + s \overset{F}{\mapsto} V + t$.[2]

To extend the notation over several iterated rounds of F, we call an $r+1$-tuple of affine subspaces $(U_1 + s_1, \ldots, U_{r+1} + s_{r+1})$ a *truncated differential trail* (over r rounds), if

$$U_i + s_i \overset{F}{\mapsto} U_{i+1} + s_{i+1} \quad \forall i \in \{1, \ldots, r+1\}.$$

Note that this definition can be generalized for subsets. For example, any impossible differential where an input difference a does not lead to an output difference b can be seen as a truncated differential with probability one as $\{0\} + a \mapsto \mathbb{F}_2^m \setminus \{b\} + 0$. However, omitting the structure (affine) subspaces provide makes it much harder to handle differentials algorithmically and we thus stick to the notation using affine subspaces. Additionally the above definition is typically extended to truncated differentials that do not hold for every x but instead with some high probability. We do not cover truncated differentials with probability smaller than one, as we only consider subspace trails that hold with probability one.

The following lemma is the key observation that links subspace trails to truncated differentials.

Lemma 1. *Given $U \overset{F}{\to} V$. Then*

$$\forall u \in U : \mathrm{Im}(\Delta_u(F)) \subseteq V.$$

Moreover, for any subspace $U \subset \mathbb{F}_2^n$ it holds that

$$U \overset{F}{\to} \mathrm{span}\left(\bigcup_{u \in U} \mathrm{Im}(\Delta_u(F)) \right)$$

Proof. Let $u \in U$. Because $U \overset{F}{\to} V$ is a subspace trail for F, for any $x \in \mathbb{F}_2^n$: both, x and $x + u$, are in a coset $U + x$ of U. Due to the subspace trail, they get mapped to a coset of V: $F(x), F(x + u) \in V + b$. Therefore their sum is again in V: $F(x) + F(x + u) \in V$. □

Lemma 1 also confirms the intuition that subspace trail attacks and truncated differentials with probability one are closely related. Grassi et al. [GRR16] already discussed this, but we want to state this explicitly:

Corollary 2 (Link between Subspace Trails and Truncated Differentials)**.** *Given $U \overset{F}{\to} V$. Then U and V determine a truncated differential with* linear *subspaces that holds with probability one: $U + 0 \overset{F}{\mapsto} V + 0$.*

Thus, while subspace trails are included in truncated differentials (as linear subspaces are a special case of affine subspaces), the converse is not true in general. In other words, using truncated differentials we obtain a bit more information on the actual structure of the investigated function. This is depicted in the following example.

[2] This notation is similar to the one for subspace trails on purpose. As we stick to subspace trails in the following sections, it is only needed here, anyway.

Example 1. We choose a single PRESENT S-box, see [Bog+07], as F and compute the subspace trail, resp. truncated differential, in terms of linear and affine subspaces for the starting difference `0x1`:

$$\{\texttt{0x0}, \texttt{0x1}\} \xrightarrow{F} \{\texttt{0x0}, \texttt{0x3}, \texttt{0x4}, \texttt{0x7}, \texttt{0x9}, \texttt{0xa}, \texttt{0xd}, \texttt{0xe}\}$$
$$\{\texttt{0x0}\} + \texttt{0x1} \xmapsto{F} \{\texttt{0x0}, \texttt{0x4}, \texttt{0xa}, \texttt{0xe}\} + \texttt{0x9}$$

The affine subspaces we obtain are one dimension smaller than the linear subspaces.

Therefore truncated differentials are rather a generalization of subspace trails than vice versa. For the remainder of this work, we focus on the search for subspace trails.

2.4 Computing A Trail for a Given Input Difference

A starting point for finding subspace trails is: Given an initial subspace, how to compute the resulting trail? One approach is based on Lemma 1. In order to compute V, we have to compute the images of the derivatives of F in direction U. To speed this up, we can exploit two facts. First, when choosing $x \in_{\mathrm{R}} \mathbb{F}_2^n$, assuming a random behavior, it is sufficient to take slightly more than n many x to compute the subspace spanned by the image. Second, we do not need to compute the image of every element in U; instead it is enough to take a basis of U, see the following lemma.

Lemma 2. *Given $U \subseteq \mathbb{F}_2^n$ and a basis $b_1, \ldots, b_k$ of U, then*

$$\operatorname{span}\left(\bigcup_{u \in U} \operatorname{Im}(\Delta_u(F))\right) = \operatorname{span}\left(\bigcup_{1 \le i \le k} \operatorname{Im}(\Delta_{b_i}(F))\right).$$

Proof. It is clear that the set on the right side is a subset of the set on the left side of the equation. Thus, we are left with showing that the left side is a subset of the right side. Moreover, as we consider the linear span on both sides, it suffices to show that any

$$v \in \bigcup_{u \in U} \operatorname{Im}(\Delta_u(F))$$

is contained in

$$\operatorname{span}\left(\bigcup_{1 \le i \le k} \operatorname{Im}(\Delta_{b_i}(F))\right).$$

We will prove this by induction over the dimension of U. The case $\dim(U) = 1$ is trivial. Now assume that the statement is correct for any U' of dimension smaller than k.

We consider

$$v \in \operatorname{Im}(\Delta_u(F))$$

for $u \in U$. That is, there exist an element $x \in \mathbb{F}_2^n$ such that

$$v = \Delta_u(F)(x) = F(x) + F(x + u).$$

As the b_i form a basis of U, we can express u as a linear combination of the b_i, that is

$$u = \sum_{i=1}^{k} \lambda_i b_i$$

for suitable $\lambda_i \in \mathbb{F}_2$. Thus

$$v = F(x) + F(x + \sum_{i=1}^{k} \lambda_i b_i).$$

By defining $x' = x + \lambda_k b_k$ we get

$$\begin{aligned} v &= F(x) + F(x + \sum_{i=1}^{k} \lambda_i b_i) \\ &= F(x' + \lambda_k b_k) + F(x' + \sum_{i=1}^{k-1} \lambda_i b_i) \\ &= F(x' + \lambda_k b_k) + F(x') + F(x') + F(x' + \sum_{i=1}^{k-1} \lambda_i b_i) \\ &= \lambda_k \Delta_{b_k}(F)(x') + \lambda' \Delta_{u'}(F)(x') \end{aligned}$$

where

$$\lambda' = \bigvee_{i=1}^{k-1} \lambda_i, \qquad u' = \sum_{i=1}^{k-1} \lambda_i b_i.$$

Thus

$$v \in \mathrm{span}(\mathrm{Im}(\Delta_{b_1}(F)) \cup \mathrm{Im}(\Delta_{u'}(F))),$$

and the lemma follows by induction as u' is contained in a $(k-1)$ dimensional subspace

$$U' = \mathrm{span}(\{b_1, \ldots, b_{k-1}\})$$

□

Assembling the above observations, we get the recursive Algorithm 1 to compute the optimal subspace trail for a given starting subspace U.

Algorithm 1 Compute subspace trails

Precondition: A nonlinear, bijective function $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ and a subspace U.

```
1 function Compute Trail(F, U)
2     if dim(U) = n then
3         return the list [U]
4     V ← ∅
5     for u_i basis vectors of U do
6         for enough x ∈_R F_2^n do
7             V ← V ∪ Δ_{u_i}(F)(x)
8     V ← span(V)
9     return the list [U] + Compute Trail(F, V)
```

Note that in the case where F has algebraic degree two, its derivative $\Delta_{u_i}(F)$ is linear in all possible directions u_i and thus we can compute V in Algorithm 1 deterministically. For the sake of simplicity, we leave out this optimization.

In Lines 6 and 7 we are sampling random elements of the subspace V. To get the full subspace we are looking for, when computing the span, we have to test enough random x. Assuming V is a random subspace of $\mathbb{F}_2^n$ and upper bounding its dimension by n, we are interested in the probability that m random vectors of length n form a matrix with full rank n. The probability for the ith vector to be linearly independent of the previous $i-1$ vectors is $1 - 2^{i-1-m}$. Thus the probability for all m vectors to be linearly independent is $\prod_{i=0}^{n-1}(1 - 2^{i-m})$. This is also known as $(2^{-m}; 2)_n$, the 2-Pochhammer symbol or 2-shifted factorial. Computing $(2^{-m}; 2)_n$ shows that it is "enough" to sample e. g. $m = n + 100$ random x to compute the full subspace V with overwhelming probability.[3]

[3] With $m = n + 100$ we obtain an error probability of 2^{-100}, and $m = n + 20$ results in roughly 2^{-20}.

Unfortunately this still does not reduce the number of possible starting differences. Thus let us now take a more detailed look at the parts of an SPN. The influence of the linear layer on subspace trails is straightforward.

Proposition 1 (Subspace trails through linear layers)**.** *Let $L : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be a bijective linear function. Then every $U \subseteq \mathbb{F}_2^n$ defines a subspace trail of the form $U \xrightarrow{L} L(U)$. U and $L(U)$ have the same dimension. Moreover, any essential subspace trail for L is of this form.*

Proof. Recall that for a subspace trail $U \xrightarrow{L} V$ it holds that

$$V \supseteq \bigcup_{u \in U} \mathrm{Im}(\Delta_u(L)).$$

For any bijective linear function

$$\mathrm{Im}(\Delta_u(L)) = \{L(x) + L(x+u)\} = \{L(x) + L(x) + L(u)\} = \{L(u)\}.$$

As L is bijective, for any pairwise different u_1, $u_2 \in U$, $L(u_1)$ and $L(u_2)$ are also different. Thus, $V = L(U)$ and $\dim(U) = \dim(V)$. □

The S-box layer exhibits a more interesting behavior. Here, we need to distinguish between S-boxes without non-trivial linear structures and ones with linear structures. The first case is covered in Section 4, the second in Section 5.

Before we study the S-box layer, we will next motivate why the problem of finding the best subspace trail is not as easy as it might seem.

3 Activating Only Single S-boxes

As mentioned in the introduction, one common heuristic used in finding long truncated differentials, is to activate only a single S-box in the input difference. Due to the close relation of truncated differentials and subspace trails discussed above it seems natural to use the same heuristic for the latter. Intuitively it may seem that using starting spaces U that contain non-zero values only for a single S-box performs the best.

In this section we argue that this is not always the best approach. This is done by considering two examples. The first one is a toy example based on the PRESENT S-box, while the second one is the cryptographic permutation KECCAK-f.

Example 2. We choose again the PRESENT S-box, $F : \mathbb{F}_2^4 \to \mathbb{F}_2^4$, and a block size of 16 bits, so the S-box layer applies four S-boxes in parallel, $F^4 : \mathbb{F}_2^{16} \to \mathbb{F}_2^{16}$. The round function $R : \mathbb{F}_2^{16} \to \mathbb{F}_2^{16}$ is then $R(x) := L \cdot F^4(x)$, where as the linear layer L we choose

$$L = L' \cdot B^{-1} = \begin{pmatrix}
0&0&0&1&1&0&1&0&0&0&0&1&1&0&1&0\\
1&0&1&1&1&0&1&1&1&0&1&1&1&0&1&1\\
0&0&0&0&0&0&0&1&1&0&1&0&1&0&1&1\\
0&0&0&1&1&0&1&1&1&0&1&0&0&0&0&0\\
1&0&1&0&0&0&0&1&1&0&1&1&1&0&1&1\\
0&0&0&1&1&0&1&1&0&0&0&0&0&0&0&1\\
0&0&0&1&0&0&0&1&1&0&1&0&1&0&1&0\\
0&0&1&1&1&0&1&1&0&0&0&0&1&0&1&0\\
1&1&1&0&0&0&0&0&1&0&1&1&0&0&0&1\\
1&0&1&1&0&0&1&0&1&0&1&0&1&0&1&0\\
1&0&1&1&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&1&0&1&0&0&0&0&0&0&0\\
1&0&1&1&1&0&1&1&0&1&0&0&1&0&1&1\\
0&0&0&0&1&0&1&0&0&0&0&0&1&0&0&0\\
1&0&1&1&0&0&0&0&0&0&0&1&1&1&1&0\\
0&0&0&1&0&0&0&1&0&0&0&0&1&0&1&0
\end{pmatrix}$$

with

$$L' = \begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1\\
0&0&0&0&0&0&0&0&0&0&1&1&1&0&1&0&1&1\\
0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&1&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1\\
0&0&0&0&0&0&0&0&0&0&1&1&0&0&1&1&1&1\\
1&0&0&0&0&0&0&0&0&0&1&0&0&1&1&0&1&1\\
0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0\\
0&0&1&0&0&0&0&0&0&0&1&1&1&1&1&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&1&1&0&1&1&1&0\\
0&0&0&0&1&0&0&0&0&0&1&0&0&1&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&1&1&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&1&1&1&1&0&1\\
0&0&0&0&0&0&0&1&0&0&1&0&1&0&1&1&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&1&1&0&1&0&1
\end{pmatrix}, \qquad B = \begin{pmatrix}
1&0&0&0&0&0&0&0&0&1&1&0&0&1&0&1&0&0\\
0&1&0&0&0&0&0&0&0&0&1&1&1&0&0&0&1&1\\
1&0&0&0&0&0&0&0&0&0&1&0&1&1&0&1&1&1\\
0&0&0&0&0&0&0&0&0&1&1&1&1&1&0&1&1&1\\
0&0&1&0&0&0&0&0&0&1&1&0&1&0&1&0&0&0\\
0&0&0&1&0&0&0&0&0&0&1&0&1&1&1&0&0&0\\
0&0&1&0&0&0&0&0&0&0&1&0&1&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&1&0&1&0&0&1&0&0\\
0&0&0&0&0&1&0&0&0&1&1&0&0&0&1&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&1&1\\
0&0&0&0&0&1&0&0&0&0&0&1&1&1&0&0&1&1\\
0&0&0&0&0&0&0&0&0&1&0&1&1&1&0&1&1&1\\
0&0&0&0&0&0&0&0&1&0&1&0&0&1&0&0&0&1\\
0&0&0&0&0&0&0&0&0&1&0&1&1&0&1&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&1&0&1&1&0&1&0&0&0
\end{pmatrix}.$$

Let us first observe which subspace trails this round function exhibits. Almost all subspace trails starting with only one active S-box are of the form

$$\dim = 1 \to \dim = 4 \to \dim = 16.$$

Thus reaching the full dimension after two rounds. In contrast to this, the subspace trail with a `0x1` difference on every S-box input, more precisely the starting subspace is $\{\texttt{0x0000}, \texttt{0x1111}\}$ – which is one dimensional but activates all four S-boxes in the beginning, has the following form:

$$\texttt{0x1111}: \quad \dim = 1 \to \dim = 9 \to \dim = 11 \to \dim = 16, \tag{5}$$

effectively lasting for a full additional round. This is of course due to the special structure of the linear layer. The left part of the chosen basis B consists of exactly the vectors spanning the space of dimension nine after the first round. In combination with the given L', all these vectors are then mapped to only activating the rightmost three S-boxes (with a `0x1` difference on the third S-box), leaving the last S-box passive.

Still, there are four subspace trails, with only one active S-box, that also reach three rounds:

$$\begin{aligned}
\texttt{0x1000}: &\quad \dim = 1 \to \dim = 3 \to \dim = 14 \to \dim = 16\\
\texttt{0x0001}: &\quad \dim = 1 \to \dim = 3 \to \dim = 15 \to \dim = 16\\
\texttt{0x0f00}: &\quad \dim = 1 \to \dim = 3 \to \dim = 14 \to \dim = 16\\
\texttt{0x000f}: &\quad \dim = 1 \to \dim = 3 \to \dim = 15 \to \dim = 16
\end{aligned}$$

But note that all these subspace trails are worse, as the last dimension, before reaching the full dimension, is higher than eleven. In this example it is thus not the best choice, to activate only one S-box. Also note that the resulting linear layer L looks sufficiently random, s. t. its hidden structure might be hard to find. Of course it is very unlikely that such an effect occurs, when the design of the linear layer follows e. g. the wide trail strategy [DR02a].

Example 3. A real-world example of the above observation is the SHA-3 hash function, more precisely Keccak-f [Ber+11; ST15]. Starting with any subspace trail of dimension one that has only one active S-box gives us a three round trail with final dimension 1139 in the best case. But instead starting with an other one dimensional subspace trail with *two* active S-boxes results in a different three round trail that has dimension 949 – a huge reduction compared to the initial trail. The exact dimensions are the following:

$$\begin{aligned}
1 \times \texttt{0x4}: &\quad \dim = 1 \to \dim = 3 \to \dim = 65 \to \dim = 1139 \to \dim = 1600\\
2 \times \texttt{0x4}: &\quad \dim = 1 \to \dim = 5 \to \dim = 49 \to \dim = \ \ 949 \to \dim = 1600
\end{aligned}$$

The big advantage of looking only at possible input spaces with one active S-box is the huge reduction in starting points we need to analyse. But as we have just seen, we

cannot rely on these special inputs, when we want to prove the absence of subspace trails. While we can argue with Corollary 1 that it is enough to look at trails starting with a one dimensional subspace, we are still facing the problem to cope with exponentially many possible subspace trails. In the next sections, we develop arguments and algorithms to again reduce this complexity.

4 S-box layers without Linear Structures

The distinction of S-boxes based on their linear structures is based on the following lemma that explains the link. Recall that $\mathrm{L}_u(F)$ is defined as the set of all α such that

$$\langle \alpha, \Delta_u(F)(x) \rangle = c \quad \forall x \in \mathbb{F}_2^n$$

Let us start by considering a single S-box $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ only.

Lemma 3. *Given an essential subspace trail $U \xrightarrow{F} V$. Then $V^\perp = \bigcap_{u \in U} \mathrm{L}_u(F)$.*

Proof. According to Lemma 1 it holds that

$$\forall x \in \mathbb{F}_2^n \ \forall u \in U : \Delta_u(F)(x) \in V$$

Now, for an arbitrary element $\alpha \in V^\perp$ this implies that

$$\forall x \in \mathbb{F}_2^n \ \forall u \in U : \langle \alpha, \Delta_u(F)(x) \rangle = 0.$$

Thus

$$V^\perp = \bigcap_{u \in U} \mathrm{L}_u(F)$$

and, as the trail is essential, the statement follows. □

In particular we have Corollary 3 as a direct consequence.

Corollary 3. *If F has no non-trivial linear structure, then there are only two essential subspace trails: $\{0\} \to \{0\}$ and $\mathbb{F}_2^n \to \mathbb{F}_2^n$.*

The most important observation is that, in the case of F without linear structures, a similar observation holds for the entire S-box layer consisting of k parallel S-boxes. Namely, in this case any essential subspace trail is *the direct product of subspace trails of the single S-box*. This implies with the Corollary 3 that any essential subspace trail through an S-box layer can simply be characterized by active and passive S-boxes. The following proposition covers exactly this.

Proposition 2 (Subspace trails through S-box layers)**.** *Let F^k be an S-box layer, and $U \xrightarrow{F^k} V$ an essential subspace trail. If F has no non-trivial linear structures, then*

$$U = V = U_1 \times \cdots \times U_k$$

where for $1 \leq i \leq k$ it holds that $U_i \in \{\{0\}, \mathbb{F}_2^n\}$.

Proof. Let $U \xrightarrow{F^k} V$ be essential. Then the following holds $\forall \alpha = (\alpha_1, \ldots, \alpha_k) \in V^\perp$:

$$\begin{aligned} \langle \alpha, \Delta_u(F^k)(x) \rangle &= \left\langle \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_k \end{pmatrix}, \begin{pmatrix} \Delta_{u_1}(F)(x_1) \\ \vdots \\ \Delta_{u_k}(F)(x_k) \end{pmatrix} \right\rangle \\ &= \sum_{i=1}^{k} \langle \alpha_i, \Delta_{u_i}(F)(x_i) \rangle = 0 \qquad (\forall x_i \in \mathbb{F}_2^n) \end{aligned}$$

Thus every term $\langle \alpha_i, \Delta_{u_i}(F)(x_i)\rangle$ is constant. As F has no non-trivial linear structures, this implies that either α_i or u_i is zero. But then, in both cases, if one of the two is zero, the other can take any value and still has to be in the corresponding subspace, because the trail is, by assumption, essential. □

It is important to note that not only Corollary 3 does not hold for F with linear structures, but more fundamentally (and maybe counterintuitively) the statement that any essential subspace trail is a direct product of subspace trails of the single S-box *does not hold in the case of F with linear structures.* A particular case of this is given by the subspace trail in Example 2 (see Eq. (5)). We come back to this case in Section 5.

Let us give a specific notation for the particular structure of the above subspaces. If $U = U_1 \times \cdots \times U_k$ with $U_i \in \{\{0\}, \mathbb{F}_2^n\}$, we write $U = \{\{0\}, \mathbb{F}_2^n\}^u$ iff $U_i = \mathbb{F}_2^n \Leftrightarrow u_i = 1$ where u_i is the ith bit of the binary depiction of u.

Now a direct consequence of the proposition is the following corollary.

Corollary 4. *If F has no non-trivial linear structures, there are only 2^k possible $U \in \{\{0\}, \mathbb{F}_2^n\}^k$, s. t. $\exists V \subseteq (\mathbb{F}_2^n)^k : U \xrightarrow{F^k} V$ is essential.*

In other words, and we would like to stress this, an S-box without non-trivial linear structures behaves like any generic S-box. Particularly the only important property of such an S-box layer is, if the S-box is "activated" or not. As, e. g. the AES S-box does not have any linear structures (and neither does its inverse has any), this means that *we cannot improve* subspace trail attacks given in [GRR17] by taking details of the S-box into account.

With the above we can construct Algorithm 2 to compute a list of all one round subspace trails. This list is enough to construct all possible subspace trails. To iterate the rounds we just have to look up the next subspace in it. After reaching a subspace with full dimension, the actual trail cannot be iterated further.

Algorithm 2 No Non-Trivial Linear Structures

Precondition: A linear layer matrix $M : \mathbb{F}_2^{n\cdot k \times n\cdot k}$.

function ONE ROUND SUBSPACE TRAILS(M)
  empty list L
  for possible initial subspaces represented by $u \in \{0,1\}^k$ **do**
    $U = \{\{0\}, \mathbb{F}_2^n\}^u$
    $U' = \{M \cdot u_i \mid u_i \text{ basis vectors of } U\}$
    $V = (V_1, \ldots, V_k) = \{\{0\}, \mathbb{F}_2^n\}^v$, s. t.

$$V_i = \begin{cases} \{0\} & \text{if } U'_i = \{0\} \\ \mathbb{F}_2^n & \text{else} \end{cases} \quad \text{resp.} \quad v_i = \begin{cases} 0 & \text{if } U'_i = \{0\} \\ 1 & \text{else} \end{cases}$$

    append (u, v) to L
  return L

Results

Besides the normal round function, we also have to take the inverse into account, as an S-box and its inverse do not necessarily have the same linear structures. It may even be the case that an S-box has no non-trivial linear structures, while its inverse has some. Examples for such S-boxes are the ones from Safer [Mas94], SC2000 [Shi+02], and Fides [Bil+13]. Moreover, the inverse of the linear layer might be weaker than the linear layer itself.

Perrin [Per17] collected a list of S-boxes used or discussed in the literature. From this list of over 200 S-boxes, we filtered those designs that are generic SPNs and applied

Table 1: Lengths of the longest subspace trails in various SPN ciphers with S-boxes that do not have any non-trivial linear structures. r_e denotes the number of encryption rounds covered, r_d the number of decryption rounds, and d the final dimension.

	Algorithm 2			
Cipher	r_e	d	r_d	d
AES [DR02b]	2	32	2	32
ANUBIS [BR]	2	104	—†	—†
KLEIN [GNL11]	3	60	2	32
KUZNYECHIK [GOST15]	1	8	1	8
PRINCE [Bor+12]	2	16	2	16
QARMA [Ava17]*	2	36	2	36

* QARMA comes with three S-boxes, two of these do not exhibit linear structures. The given results are thus for QARMA instantiated with either σ_1, or σ_2.
† ANUBIS uses involutional operations, so we only give the encryption trails.

Algorithm 2 to them, if the corresponding S-box has no non-trivial linear structures. The resulting lengths of the longest subspace trails are listed in Table 1. To the best of our knowlegde, all these trails are not better than what is already known in the literature.

Note that this algorithm works for any generic S-box that has no non-trivial linear structures. Regarding its runtime, there are 2^k possible initial subspaces. For each we have to compute the corresponding output subspace, which boils down to a matrix multiplication with every basis vector of the input subspace. Altogether this results in an overall runtime that is exponential in the number of S-boxes and linear in the dimension of the linear layer. A non-optimized C implementation takes seconds to finish the computations for the tested ciphers on a single CPU core of a standard laptop.

From a designers perspective this maintains the ability to independently choose the linear and S-box layer, with respect to subspace trails. Nevertheless, we might not want to use such a decoupled design approach, e. g. in the design process of lightweight cryptography, we often aim for vigorous optimized designs. During such an optimization process, we might decide to trade strong security properties for more efficient parts by carefully matching the components of the whole design. A recent proposal, which does exactly this, is [Ban+17].

Thus we also have to cover the other case, involving S-boxes with non-trivial linear structures.

5 S-box layers with Linear Structures

As mentioned above, the core observation in the case of an S-box layer with an S-box without linear structures is that any essential subspace trail is the direct product of subspace trails for the single S-boxes. Unfortunately, this does not hold when moving to S-boxes that actually do have linear structures. An example of this has been presented in Section 4.

However, without a significant reduction of the possible starting subspaces that have to be considered, we do not have the possibility to guarantee the non-existence of long subspace trails. To overcome this problem, we move from considering the input subspace to the first S-box layer to the output subspace of the first S-box layer. The key point

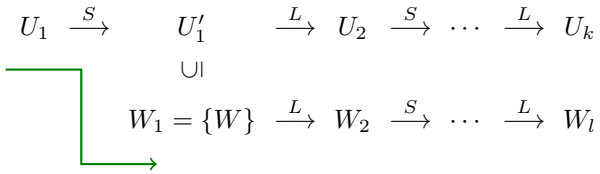

Figure 2: How Corollary 1 is used to upper bound the length of subspace trails.

is that even though we cannot restrict the input subspaces in a meaningful way, we can actually do so for the output subspaces. Namely, we will show below that the output space of the first S-box layer always contains a subspace that (again) activates only one S-box. Figure 2 depicts this approach.

More precisely, as it is too expensive to check the trail length of every possible U_1, we look at the output spaces U_1' after the first S-box layer. Applying Corollary 1 results still in a subspace trail starting from $W_1 \subseteq U_1'$, and we additionally restrict ourselves to one dimensional W_1. Eventually, we use the length l of the subspace trail $W_1 \to W_l$ to provably bound the length k of the longest subspace trail. Note that it stays uncertain at this point if any of those trails can actually be extended backwards through the first S-box layer, thus $k = l$ if the trail cannot be backward extended, or $k = l + 1$ else. However, using this approach we are able to provably bound the longest subspace trail. Moreover our bound is either tight or off by at most one round.

The remaining problems are: What is the problem with backward extension? Why does the length of $W_1 \to W_l$ actually gives a meaningful bound on the longest subspace trail? And third, we have to find a (small) set $\mathbb{W}$ of all possible W_1, s. t. *every possible* U_1 contains at least one element from it after the first S-box layer.

Backward extension Admittedly it is easy, given any W_1, to find an U_1, s. t. U_1 is a starting point for a subspace trail that contains W_1 after the S-box layer. But the real problem is that we want to backward extend W_1 in such a way that both trails continue with the same dimensions. This can either happen if $U_1 \to W_1$ is essential and $\dim(U_1) = \dim(W_1) = 1$, but then U_1 would be a probability one differential characteristic for the S-box layer, or if $W_1 \to W_2$ is not essential. In the latter case, it is not clear how to extend W_1 backwards, as we are missing the information what else U_1' contains and is missing in W_1 to make the trail essential.

Bounding the length Assume the longest subspace trail starting from an element in $\mathbb{W}$ covers l rounds. From Fig. 2 it is clear that $l \geq k$, and we can either backward extend W_1 or we cannot. If $l = k$ and backward extension is possible, our bound l is off by one round compared to the actual longest subspace trail. On the other side, if we cannot backward extend the trail, our bound is tight. If $l > k$ (this is possible, because W_1 is a subset of U_1' and thus might cover more rounds), we effectively found a longer trail, we can directly use this as the longest subspace trail, and our bound is tight.

The set $\mathbb{W}$ For $\mathbb{W}$, we look at all possible ways to activate one S-box after the first S-box layer. As the choices for activating one S-box are very limited and typically the number of S-boxes is small compared to the overall block size, $\mathbb{W}$ is also small compared to all possible starting points. Thus we need to show that $\mathbb{W}$ is an appropriate set in the sense of our above requirement – that is, for every possible U_1 there is one element W, s. t. $W \in U_1'$ after the first S-box layer. Before stating the main proposition, we need the following lemma.

Lemma 4. *Let F be an S-box with differential uniformity $\delta_F < 2^n$, i. e. for all $\alpha, \beta \in \mathbb{F}_2^n$, the cardinality of $\{x \in \mathbb{F}_2^n \mid F(x) + F(x+\alpha) = \beta\}$ is smaller than 2^n. Then*

$$\forall u \in \mathbb{F}_2^n \setminus \{0\} : \mathrm{L}_u(F) \neq \mathbb{F}_2^n.$$

Proof. Assume $\delta_F < 2^n$ and there is an u, s. t. $\mathrm{L}_u(F) = \mathbb{F}_2^n$. Let $\{e_i\}$ be the canonical basis for $\mathrm{L}_u(F)$. For every e_i the following holds:

$$\langle e_i, \Delta_u(F)(x) \rangle = c_{e_i} \qquad \forall x \in \mathbb{F}_2^n, c_{e_i} \in \mathbb{F}_2$$

But then the differential characteristic $u = (u_1, \ldots, u_n)$ to $v = (c_{e_1}, \ldots, c_{e_n})$ holds with probability one and thus $\delta_F = 2^n$, which is a contradiction. □

We construct the set $\mathbb{W}$ of subspaces as follows:

$$\mathbb{W} \coloneqq \left\{ W_{i,\alpha} \coloneqq \{0\}^{i-1} \times \{0, \alpha\} \times \{0\}^{k-i} \;\middle|\; \alpha \in \mathbb{F}_2^n, 1 \leq i \leq k \right\}.$$

The following proposition shows that any output subspace after the S-box layer actually has to contain at least one element $W \in \mathbb{W}$.

Proposition 3. *Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be an S-box with differential uniformity $\delta_F < 2^n$, and $U \xrightarrow{F^k} V$ a non-trivial subspace trail. Then there exists a $W \in \mathbb{W}$ such that*

$$W \subseteq V$$

Proof. As U is non-trivial, it contains at least one non-zero element $u = (u_1, \ldots, u_k)$. For an arbitrary element $\beta \in V^\perp$ and all $x \in \mathbb{F}_2^n$ we get

$$\langle \beta, \Delta_u(F)(x) \rangle = 0$$

which implies that $\beta \in L_u^0(F^k)$ and thus

$$V^\perp \subseteq L_u^0(F^k)$$

Furthermore,

$$\mathrm{L}_u^0(F^k) \subseteq \mathrm{L}_{u_1}(F) \times \mathrm{L}_{u_2}(F) \times \cdots \times \mathrm{L}_{u_k}(F).$$

As u is non-zero, at least one of its components u_i is non-zero. According to Lemma 4, $\mathrm{L}_{u_i}(F) \neq \mathbb{F}_2^n$, which implies that there exists an $\alpha \in \mathbb{F}_2^n$ such that

$$\mathrm{L}_{u_i} \subseteq \{0, \alpha\}^\perp$$

Putting things together we conclude that

$$V^\perp \subseteq L_u^0(F^k) \subseteq (\mathbb{F}_2^n)^{i-1} \times \{0, \alpha\}^\perp \times (\mathbb{F}_2^n)^{k-i}.$$

By looking at the orthogonal complement of both sides we finally get

$$\left((\mathbb{F}_2^n)^{i-1} \times \{0, \alpha\}^\perp \times (\mathbb{F}_2^n)^{k-i} \right)^\perp = \{0\}^{i-1} \times \{0, \alpha\} \times \{0\}^{k-i} \subseteq V,$$

which concludes the proof. □

We can now build Algorithm 3, which simply applies the algorithm in Section 2.4 to every element in $\mathbb{W}$ to complete the corresponding trail. Because $|\mathbb{W}|$ is small and Algorithm 1 is efficient, this will allow us to compute the longest subspace trail that starts after the first layer of S-boxes, and its trail bounds the longest possible subspace trail.

Algorithm 3 Generic Subspace Trail Search

Precondition: A linear layer matrix $M : \mathbb{F}_2^{n \cdot k \times n \cdot k}$, and an S-box $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$.

```
function GENERIC SUBSPACE TRAIL LENGTH(M, F)
    empty list L
    for possible initial subspaces represented by W_{i,α} ∈ 𝕎 do
        L.append(COMPUTE TRAIL(F^k ∘ M, ({0}, W_{i,α})))
    return max{len(t) | t ∈ L}
```

Table 2: Lengths of the longest subspace trails in various SPN ciphers with 1-linear structures. r_e denotes the number of rounds covered for encryption, r_d the number of rounds covered for decryption, and d the final dimension.
Algorithms 2 and 3 are compared, where the algorithm taking the influence of linear structures into account (Algorithm 3) finds better results, that is longer trails and/or lower dimensions.

	Algorithm 3				Algorithm 2			
Cipher	r_e	d	r_d	d	r_e	d	r_d	d
ASCON [Dob+16]	3	298	1	125	3	310	1	155
GIFT [Ban+17]	3	60	3	60	2	16	2	16
KECCAK [Ber+11]	2	546	1	169	2	1290	1	270
PRESENT [Bog+07]	3	43	3	63	2	16	2	16
PRIDE [Alb+14]	2	31	2	34	2	56	1	40
QARMA [Ava17]*	2	36	2	36	2	36	2	36
SERPENT [BAK98]	2	88	2	62	2	100	2	68
SKINNY64 [Bei+16]	5	48	5	48	4	48	4	48
SKINNY128 [Bei+16]	5	96	5	96	5	96	5	96

* The given results are for QARMA instantiated with σ_0.

Results

We again filtered the list of S-boxes by Perrin [Per17] for SPN constructions and S-boxes that exhibit 1-linear structures. The runtime of Algorithm 3 is similar to the one of the previous algorithm. Here, we have to compute trails for $|\mathbb{W}| = k \cdot 2^n$ many initial subspaces. Computing trails for a given U_0 is dominated by Gaussian eliminations for the basis reductions of the corresponding subspaces. Thus, the runtime mainly depends on the block size, length of trails and number and size of the S-box. Although our reference C implementation has room for optimizations, it takes only a few seconds to find the trails of 64-bit block ciphers and a few minutes to find the trails of 128-bit block ciphers using a single CPU core on a standard laptop.

In Table 2, we upper bound the length of subspace trails for several SPN ciphers. To the best of our knowledge, no subspace trails for any of these ciphers is known in the literature at the time of writing. We give the resulting trail lengths of Algorithm 3 with the corresponding dimensions of the final subspace. Note that for a subspace trail of length r the actual upper bound is $r + 1$. But as already mentioned above, we might be able to backward extend the trail for one more round.

One example, where exactly this happens, is SKINNY128. The best impossible differential known is given in [Bei+16, Section 4.3] and exploits a six round differential in the encryption direction that misses another five round differential in decryption direction.

Algorithm 3 finds a five round subspace trail, thus in this case the known differential reaches the $5+1$ bound.

As a comparison to Algorithm 2, we also give the resulting longest subspace trails for the case of an S-box without linear structures in the second part of Table 2. Here again SKINNY64 is a nice example, because it highlights the influence of the linear structures in its S-box. The bound of four rounds by Algorithm 2 is too low compared to the actual best trail.

6 Open Problems

We presented efficient algorithms to identify (or bound the length of) subspace trails for any SPN. We like to mention some interesting possibilities for future research.

There are basically two directions to generalize our results. First, it would be interesting to not only focus on SPN ciphers but also cover Feistel structures. A technically related problem to the generalization is the following: For a subspace trail $U \xrightarrow{F^k} V$ the joint space of inputs and outputs $U \times V \subseteq \mathbb{F}_2^{2n}$ is (obviously) a direct product of subspaces. A question that appears when studying Feistel structures, is if there exists a meaningful generalization where one considers more general subspaces of $\mathbb{F}_2^{2n}$ for the joint input and output spaces. An answer to this question could not only be useful for the case of Feistel ciphers, but be of independent interest as well. However, although our results for SPN ciphers do not directly apply to Feistel ciphers, our algorithms can be used to obtain probability one subspace trails for them. We observe that the longest subspace trails our algorithms can find are compatible with the longest probability one truncated differentials that are used in the literature for obtaining impossible differentials via the miss-in-the-middle technique. Moreover, since we take into account the S-box properties and we focus on the dimensions instead of non-indeterminate bit differences, the dimensions we find are most of the time better than the ones used in the literature. These reductions in the dimensions can provide more freedom to the attacker and may lead to better attacks. We provide the longest subspace trails that our search algorithms find in Table 3.

To obtain our results, we search for the longest subspace trail starting with a single active S-box only. For ciphers like SIMON that do not use an S-box, we consider differences in one word only. Although we obtain a 13-round subspace trail for SKIPJACK, we cannot use it to extend the 24-round impossible differential of [BBS05; BBS99a] to 25 or 26 rounds, because SKIPJACK does not use identical round functions for all rounds. The limitations of our algorithms should also be considered when constructing impossible differentials using probability one subspace trails:

- A subspace trail with full dimension may still be useful for constructing impossible differentials via the miss-in-the-middle technique. Even if it reaches full dimension, it may still contain exploitable information like, e. g., an S-box output difference being nonzero.
- Unlike SPN, for Feistel ciphers it may be possible to obtain $(r_e + r_d + 1)$-round impossible differentials (instead of $r_e + r_d$) by using properties of the round function.
- This direct application of our search algorithms to Feistel ciphers of course do not prove the resistance against subspace trail attacks.

The second way to generalize our results is to consider truncated differential trails with probability one instead of subspace trails. As mentioned in Section 2, the only difference is that one moves from subspaces to affine subspaces. Indeed, in the case of S-boxes without linear structures, our results are directly applicable to truncated differential trails as well. However, in the case of S-boxes with linear structures, the situation is quite different. Here,

Table 3: Lengths of the longest truncated differentials used in the best known impossible differentials in various Feistel ciphers. r denotes the number of rounds covered by the differential, and d the dimension at the miss-in-the-middle-point. Our results are obtained via subspace trails.

	Literature			Our Results	
Cipher	r	d	Source	r	d
Camellia	4	128†	[BL12]	4	105
Clefia	5	128†	[Tsu+08]	5	112
Lblock	7	—‡	[WZ11]	7	58
Piccolo	4	64†	[WW12]	3	48
Simon32/64	6	29	[DF16]	6	30*
Simon64/128	8	—‡	[BNS14]	8	62
Simon128/256	12	—‡	[BNS14]	12	126
Skipjack	12	48	[BBS05; BBS99a]	13	48

* Moving to affine subspaces reduces this dimension to 29. Thus, this is an another example to show the differences between subspace trails and truncated differentials.
† The contradiction for the miss-in-the-middle is caused by exploiting special properties of the round function.
‡ The truncated differential is not explicitly given in the literature, so we do not know its exact dimension.

the main task is to find a suitable reduction in the possible offsets of the affine spaces used, which seems non-trivial.

Finally, we like to mention that invariant subspaces (see e. g. [LMR15]) behave very differently, even though the names might suggest the opposite. Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$. A subspace $U \subseteq \mathbb{F}_2^n$ is called *invariant with respect to* F, if

$$\exists a, b : F(U + a) = U + b$$

The important difference here is that invariant subspaces are defined for *one* coset $U + a$ being mapped to one other coset $U + b$. In contrast, for a subspace trail *all* cosets of U have to be mapped to cosets of V.

It is actually not trivial to understand the possible invariant subspaces of an S-box layer, even in the case of an S-box without linear structures (see [LR17] for interesting partial results along those lines). In particular, it is not the case that an invariant subspace for an S-box layer is always the direct product of invariant subspaces of the single S-boxes. An easy counterexample is the following: Consider two parallel application of an S-box. If the two inputs are identical, so are the outputs. That is, the space

$$U = \{x, x\}$$

is an invariant subspace for any S-box and is clearly not a direct product. Hence, an efficient classification of all invariant subspaces for an S-box layer is a non-trivial but interesting open problem as well.

Acknowledgments

The experimental results were partially computed on the Ruhr-University's "Crypto Crunching Cluster" (C3). This work was supported by the German Research Foundation through

the DFG Research Training Group GRK 1817/1 UbiCrypt. Cihangir Tezcan was financially supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) BIDEB-2219 Postdoctoral Research program with Project No. 1059B191600050.

References

[Alb+14] Martin R. Albrecht, Benedikt Driessen, Elif Bilge Kavun, Gregor Leander, Christof Paar, and Tolga Yalçin. "Block Ciphers - Focus on the Linear Layer (feat. PRIDE)." In: *CRYPTO 2014, Part I*. Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8616. LNCS. Springer, Heidelberg, Aug. 2014, pp. 57–76. DOI: 10.1007/978-3-662-44371-2_4.

[Ava17] Roberto Avanzi. "The QARMA Block Cipher Family." In: *IACR Trans. Symm. Cryptol.* 2017.1 (2017), pp. 4–44. ISSN: 2519-173X. DOI: 10.13154/tosc.v2017.i1.4-44.

[BAK98] Eli Biham, Ross J. Anderson, and Lars R. Knudsen. "Serpent: A New Block Cipher Proposal." In: *FSE'98*. Ed. by Serge Vaudenay. Vol. 1372. LNCS. Springer, Heidelberg, Mar. 1998, pp. 222–238. DOI: 10.1007/3-540-69710-1_15.

[BBS05] Eli Biham, Alex Biryukov, and Adi Shamir. "Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials." In: *Journal of Cryptology* 18.4 (Sept. 2005), pp. 291–311. DOI: 10.1007/s00145-005-0129-3.

[BBS99a] Eli Biham, Alex Biryukov, and Adi Shamir. "Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials." In: *EUROCRYPT'99*. Ed. by Jacques Stern. Vol. 1592. LNCS. Springer, Heidelberg, May 1999, pp. 12–23. DOI: 10.1007/3-540-48910-X_2.

[BBS99b] Eli Biham, Alex Biryukov, and Adi Shamir. "Miss in the Middle Attacks on IDEA and Khufu." In: *FSE'99*. Ed. by Lars R. Knudsen. Vol. 1636. LNCS. Springer, Heidelberg, Mar. 1999, pp. 124–138. DOI: 10.1007/3-540-48519-8_10.

[BL12] Dongxia Bai and Leibo Li. "New Impossible Differential Attacks on Camellia." In: *ISPEC*. Vol. 7232. LNCS. Springer, 2012, pp. 80–96. DOI: 10.1007/978-3-642-29101-2_6.

[BLN15] Céline Blondeau, Gregor Leander, and Kaisa Nyberg. "Differential-Linear Cryptanalysis Revisited." In: *FSE 2014*. Ed. by Carlos Cid and Christian Rechberger. Vol. 8540. LNCS. Springer, Heidelberg, Mar. 2015, pp. 411–430. DOI: 10.1007/978-3-662-46706-0_21.

[BLN16] Céline Blondeau, Gregor Leander, and Kaisa Nyberg. "Differential-Linear Cryptanalysis Revisited." In: *Journal of Cryptology* (Oct. 2016), pp. 1–30. ISSN: 1432-1378. DOI: 10.1007/s00145-016-9237-5.

[BN14] Céline Blondeau and Kaisa Nyberg. "Links between Truncated Differential and Multidimensional Linear Properties of Block Ciphers and Underlying Attack Complexities." In: *EUROCRYPT 2014*. Ed. by Phong Q. Nguyen and Elisabeth Oswald. Vol. 8441. LNCS. Springer, Heidelberg, May 2014, pp. 165–182. DOI: 10.1007/978-3-642-55220-5_10.

[BNS14] Christina Boura, María Naya-Plasencia, and Valentin Suder. "Scrutinizing and Improving Impossible Differential Attacks: Applications to CLEFIA, Camellia, LBlock and Simon." In: *ASIACRYPT 2014, Part I*. Ed. by Palash Sarkar and Tetsu Iwata. Vol. 8873. LNCS. Springer, Heidelberg, Dec. 2014, pp. 179–199. DOI: 10.1007/978-3-662-45611-8_10.

[BR] Paulo Barreto and Vincent Rijmen. *The ANUBIS Block Cipher*. First Open NESSIE Workshop.

[Ban+17] Subhadeep Banik, Sumit Kumar Pandey, Thomas Peyrin, Siang Meng Sim, Yosuke Todo, and Yu Sasaki. "GIFT: A Small Present." In: *CHES 2017*. LNCS. To appear, available at https://eprint.iacr.org/2017/622. Springer, 2017.

[Bei+16] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. "The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS." In: *CRYPTO 2016, Part II*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9815. LNCS. Springer, Heidelberg, Aug. 2016, pp. 123–153. DOI: 10.1007/978-3-662-53008-5_5.

[Ber+11] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. *The Keccak reference*. Available at http://keccak.noekeon.org/. Jan. 2011.

[Bil+13] Begül Bilgin, Andrey Bogdanov, Miroslav Knežević, Florian Mendel, and Qingju Wang. "Fides: Lightweight Authenticated Cipher with Side-Channel Resistance for Constrained Hardware." In: *CHES 2013*. Ed. by Guido Bertoni and Jean-Sébastien Coron. Vol. 8086. LNCS. Springer, Heidelberg, Aug. 2013, pp. 142–158. DOI: 10.1007/978-3-642-40349-1_9.

[Bog+07] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. "PRESENT: An Ultra-Lightweight Block Cipher." In: *CHES 2007*. Ed. by Pascal Paillier and Ingrid Verbauwhede. Vol. 4727. LNCS. Springer, Heidelberg, Sept. 2007, pp. 450–466. DOI: 10.1007/978-3-540-74735-2_31.

[Bor+12] Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knežević, Lars R. Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, Peter Rombouts, Søren S. Thomsen, and Tolga Yalçin. "PRINCE - A Low-Latency Block Cipher for Pervasive Computing Applications - Extended Abstract." In: *ASIACRYPT 2012*. Ed. by Xiaoyun Wang and Kazue Sako. Vol. 7658. LNCS. Springer, Heidelberg, Dec. 2012, pp. 208–225. DOI: 10.1007/978-3-642-34961-4_14.

[CV05] Anne Canteaut and Marion Videau. "Symmetric Boolean functions." In: *IEEE Trans. Information Theory* 51.8 (2005), pp. 2791–2811. DOI: 10.1109/TIT.2005.851743.

[Cui+17] Ting Cui, Chenhui Jin, Bin Zhang, Zhuo Chen, and Guoshuang Zhang. "Searching all truncated impossible differentials in SPN." In: *IET Information Security* 11.2 (2017), pp. 89–96. DOI: 10.1049/iet-ifs.2015.0052.

[DF16] Patrick Derbez and Pierre-Alain Fouque. "Automatic Search of Meet-in-the-Middle and Impossible Differential Attacks." In: *CRYPTO 2016, Part II*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9815. LNCS. Springer, Heidelberg, Aug. 2016, pp. 157–184. DOI: 10.1007/978-3-662-53008-5_6.

[DR02a] Joan Daemen and Vincent Rijmen. "AES and the Wide Trail Design Strategy (Invited Talk)." In: *EUROCRYPT 2002*. Ed. by Lars R. Knudsen. Vol. 2332. LNCS. Springer, Heidelberg, 2002, pp. 108–109. DOI: 10.1007/3-540-46035-7_7.

[DR02b] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002. ISBN: 3-540-42580-2. DOI: 10.1007/978-3-662-04722-4.

[Dob+16] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. *Ascon v1. 2*. Submission to the CAESAR Competition. Available at http://ascon.iaik.tugraz.at/files/asconv12.pdf. 2016.

[Eve88] Jan-Hendrik Evertse. "Linear Structures in Blockciphers." In: *EUROCRYPT'87*. Ed. by David Chaum and Wyn L. Price. Vol. 304. LNCS. Springer, Heidelberg, Apr. 1988, pp. 249–266. DOI: 10.1007/3-540-39118-5_23.

[GNL11] Zheng Gong, Svetla Nikova, and Yee Wei Law. "KLEIN: A new family of lightweight block ciphers." In: *RFIDSec'11*. Ed. by Ari Juels and Christof Paar. Vol. 7055. LNCS. Springer, Heidelberg, 2011, pp. 1–18.

[GOST15] Federal Agency on Technical Regulation and Metrology (GOST). *(GOST R 34.12-2015) Information Technology – Cryptographic Data Security – Block Ciphers*. 2015.

[GR16] Lorenzo Grassi and Christian Rechberger. "Practical Low Data-Complexity Subspace-Trail Cryptanalysis of Round-Reduced PRINCE." In: *INDOCRYPT 2016*. Ed. by Orr Dunkelman and Somitra Kumar Sanadhya. Vol. 10095. LNCS. Springer, Heidelberg, Dec. 2016, pp. 322–342. DOI: 10.1007/978-3-319-49890-4_18.

[GRR16] Lorenzo Grassi, Christian Rechberger, and Sondre Rønjom. "Subspace Trail Cryptanalysis and its Applications to AES." In: *IACR Trans. Symm. Cryptol.* 2016.2 (2016). http://tosc.iacr.org/index.php/ToSC/article/view/571, pp. 192–225. ISSN: 2519-173X. DOI: 10.13154/tosc.v2016.i2.192-225.

[GRR17] Lorenzo Grassi, Christian Rechberger, and Sondre Rønjom. "A New Structural-Differential Property of 5-Round AES." In: *EUROCRYPT 2017, Part II*. Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10211. LNCS. Springer, Heidelberg, 2017, pp. 289–317. DOI: 10.1007/978-3-319-56614-6_10.

[Gra17] Lorenzo Grassi. *Structural Truncated Differential Attacks on round-reduced AES*. Cryptology ePrint Archive, Report 2017/832. http://eprint.iacr.org/2017/832. 2017.

[KRW99] Lars R. Knudsen, Matthew J. B. Robshaw, and David Wagner. "Truncated Differentials and Skipjack." In: *CRYPTO'99*. Ed. by Michael J. Wiener. Vol. 1666. LNCS. Springer, Heidelberg, Aug. 1999, pp. 165–180. DOI: 10.1007/3-540-48405-1_11.

[Kim+03] Jongsung Kim, Seokhie Hong, Jaechul Sung, Changhoon Lee, and Sangjin Lee. "Impossible Differential Cryptanalysis for Block Cipher Structures." In: *INDOCRYPT 2003*. Ed. by Thomas Johansson and Subhamoy Maitra. Vol. 2904. LNCS. Springer, Heidelberg, Dec. 2003, pp. 82–96.

[Knu95] Lars R. Knudsen. "Truncated and Higher Order Differentials." In: *FSE'94*. Ed. by Bart Preneel. Vol. 1008. LNCS. Springer, Heidelberg, Dec. 1995, pp. 196–211. DOI: 10.1007/3-540-60590-8_16.

[Knu98] Lars Knudsen. *DEAL - A 128-bit Block Cipher*. AES Submission. 1998.

[LMR15] Gregor Leander, Brice Minaud, and Sondre Rønjom. "A Generic Approach to Invariant Subspace Attacks: Cryptanalysis of Robin, iSCREAM and Zorro." In: *EUROCRYPT 2015, Part I*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9056. LNCS. Springer, Heidelberg, Apr. 2015, pp. 254–283. DOI: 10.1007/978-3-662-46800-5_11.

[LR17] Yunwen Liu and Vincent Rijmen. *New Observations on Invariant Subspace Attack*. Cryptology ePrint Archive, Report 2017/278. http://eprint.iacr.org/2017/278. 2017.

[Lai95] Xuejia Lai. "Additive and Linear Structures of Cryptographic Functions." In: *FSE'94*. Ed. by Bart Preneel. Vol. 1008. LNCS. Springer, Heidelberg, Dec. 1995, pp. 75–85. DOI: 10.1007/3-540-60590-8_6.

[Luo+09] Yiyuan Luo, Zhongming Wu, Xuejia Lai, and Guang Gong. *A Unified Method for Finding Impossible Differentials of Block Cipher Structures*. Cryptology ePrint Archive, Report 2009/627. http://eprint.iacr.org/2009/627. 2009.

[Luo+14] Yiyuan Luo, Xuejia Lai, Zhongming Wu, and Guang Gong. "A unified method for finding impossible differentials of block cipher structures." In: *Inf. Sci.* 263 (2014), pp. 211–220. DOI: 10.1016/j.ins.2013.08.051.

[MT14] Rusydi H. Makarim and Cihangir Tezcan. "Relating Undisturbed Bits to Other Properties of Substitution Boxes." In: *LightSec 2014*. Ed. by Thomas Eisenbarth and Erdinç Öztürk. Vol. 8898. LNCS. Springer, 2014, pp. 109–125. DOI: 10.1007/978-3-319-16363-5_7.

[Mas94] James L. Massey. "SAFER K-64: A Byte-Oriented Block-Ciphering Algorithm." In: *FSE'93*. Ed. by Ross J. Anderson. Vol. 809. LNCS. Springer, Heidelberg, Dec. 1994, pp. 1–17. DOI: 10.1007/3-540-58108-1_1.

[Per17] Léo Paul Perrin. "Cryptanalysis, Reverse-Engineering and Design of Symmetric Cryptographic Algorithms." Available at http://orbilu.uni.lu/bitstream/10993/31195/1/thesis.pdf. PhD thesis. University of Luxembourg, 2017.

[ST15] National Institute of Standards and Technology. *FIPS PUB 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. Aug. 2015. DOI: 10.6028/NIST.FIPS.202.

[ST17] Yu Sasaki and Yosuke Todo. "New Impossible Differential Search Tool from Design and Cryptanalysis Aspects - Revealing Structural Properties of Several Ciphers." In: *EUROCRYPT 2017, Part III*. Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10212. LNCS. Springer, Heidelberg, 2017, pp. 185–215. DOI: 10.1007/978-3-319-56617-7_7.

[Shi+02] Takeshi Shimoyama, Hitoshi Yanami, Kazuhiro Yokoyama, Masahiko Takenaka, Kouichi Itoh, Jun Yajima, Naoya Torii, and Hidema Tanaka. "The Block Cipher SC2000." In: *FSE 2001*. Ed. by Mitsuru Matsui. Vol. 2355. LNCS. Springer, Heidelberg, Apr. 2002, pp. 312–327. DOI: 10.1007/3-540-45473-X_26.

[Sun+16] Bing Sun, Meicheng Liu, Jian Guo, Vincent Rijmen, and Ruilin Li. "Provable Security Evaluation of Structures Against Impossible Differential and Zero Correlation Linear Cryptanalysis." In: *EUROCRYPT 2016, Part I*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9665. LNCS. Springer, Heidelberg, May 2016, pp. 196–213. DOI: 10.1007/978-3-662-49890-3_8.

[TTD14] Cihangir Tezcan, Halil Kemal Taskin, and Murat Demircioglu. "Improbable Differential Attacks on Serpent using Undisturbed Bits." In: *SIN*. Ed. by Ron Poet and Muttukrishnan Rajarajan. ACM, 2014, p. 145. DOI: 10.1145/2659651.2659660.

[Tez14] Cihangir Tezcan. "Improbable differential attacks on Present using undisturbed bits." In: *J. Computational Applied Mathematics* 259 (2014), pp. 503–511.

[Tez16] Cihangir Tezcan. "Truncated, Impossible, and Improbable Differential Analysis of ASCON." In: *ICISSP 2016*. Ed. by Olivier Camp, Steven Furnell, and Paolo Mori. SciTePress, 2016, pp. 325–332. DOI: 10.5220/0005689903250332.

[Tsu+08] Yukiyasu Tsunoo, Etsuko Tsujihara, Maki Shigeri, Teruo Saito, Tomoyasu Suzaki, and Hiroyasu Kubo. "Impossible Differential Cryptanalysis of CLEFIA." In: *FSE 2008*. Ed. by Kaisa Nyberg. Vol. 5086. LNCS. Springer, Heidelberg, Feb. 2008, pp. 398–411. DOI: 10.1007/978-3-540-71039-4_25.

[WJ17] Qian Wang and Chenhui Jin. "Upper bound of the length of truncated impossible differentials for AES." In: *Des. Codes Cryptogr.* (2017). DOI: 10.1007/s10623-017-0411-z.

[WW12] Shengbao Wu and Mingsheng Wang. "Automatic Search of Truncated Impossible Differentials for Word-Oriented Block Ciphers." In: *INDOCRYPT 2012*. Ed. by Steven D. Galbraith and Mridul Nandi. Vol. 7668. LNCS. Springer, Heidelberg, Dec. 2012, pp. 283–302. DOI: 10.1007/978-3-642-34931-7_17.

[WZ11] Wenling Wu and Lei Zhang. "LBlock: A Lightweight Block Cipher." In: *ACNS 11*. Ed. by Javier Lopez and Gene Tsudik. Vol. 6715. LNCS. Springer, Heidelberg, June 2011, pp. 327–344. DOI: 10.1007/978-3-642-21554-4_19.

A Sage code

```
def to_n_bits(n, x):
    return Integer(x).digits(base=2, padto=n)[::-1]
```

```
def derivative(f, u):
    return lambda x: f(x) + f(x + u)
```

```
def active_sboxes_to_subspace(bits, n):
    """
    Return a subspace of dimension n*k, corresponding
    to full spaces of dim. n where index in bits is one
    and to zero spaces of dim. n where the corresponding
    index is zero and k = len(bits).
    """
    vs = VectorSpace(GF(2), n)
    zero_space = vs.subspace([])
    full_space = vs.subspace(identity_matrix(n))
    ls = [zero_space if i == 0 else full_space
          for i in bits]
    return reduce(lambda a, b: a.direct_sum(b), ls)
```

```
def compute_trail(f, U, n):
    """
    Return the subspace trail from U onwards

    INPUT:
        - ``f`` -- function; map from GF(2)^n to GF(2)^n
        - ``U`` -- subspace; defining the starting point
        - ``n`` -- integer; dimension of the vector space
    """
    if U.dimension() == n:
        return [U]

    vs = VectorSpace(GF(2), n)
    V = []
    for _ in range(int(1.5*n)):
        V += [derivative(f, u)(vs.random_element())
              for u in U.basis() + [vs.zero()]]
    V = vs.subspace(V)

    return [U] + compute_trail(f, V, n)
```

```
def one_round_trails(linear_layer, k):
    """
    Return a list of subspace pairs (U, V), st U and V
    form a subspace trail through k parallel applications
    of an Sbox without linear structures followed by the
    given linear layer
```

```
7
8       INPUT:
9           - ``linear_layer`` -- matrix; a n*k `times` n*k
10                                matrix over GF(2)
11          - ``k`` -- integer; the number of parallel SBoxes
12      """
13      n = Integer(linear_layer.ncols() / k)
14
15      # we have to check 2^k possible initial
16      # U = [(u_1, u_2, ..., u_k)], u_i \in {0, 1}
17      subspaces = {}
18      for u in range(1, 1<<k):
19          # compute U from active SBoxes
20          u_bits = Integer(u).digits(base=2, padto=k)
21          U = active_sboxes_to_subspace(u_bits, n)
22
23          # map linear layer over basis vectors
24          v_basis = [linear_layer * bi for bi in U.basis()]
25
26          # reduce basis to one vector that has a one entry
27          # iff at least one of the basis vectors has a one
28          # entry at the same position
29          v_bits = map(lambda bi:
30              reduce(lambda a, b:
31                  int(a) | int(b),
32                  bi),
33              zip(*v_basis))
34
35          # reduce bits to one per sbox only
36          v_bits = [1 if v_bits[i*n:(i+1)*n] != [0]*n else 0
37                    for i in range(k)]
38
39          # compute V from active SBoxes
40          V = active_sboxes_to_subspace(v_bits, n)
41          subspaces[U] = V
42      return subspaces


1   def algorithm3(f, k, n):
2       """
3       Return the set of all subspace trails containing W_{i,alpha}
4
5       INPUT:
6           - ''f'' -- function; mapping from F_2^n -> F_2^n
7           - ''k'' -- integer; number of parallel S-boxes in f
8           - ''n'' -- integer; dimension of one S-box
9       """
10      from itertools import product
11
12      dim = n*k
13      vs = VectorSpace(GF(2), dim)
14
15      trails = [[]]
```

```
16      # simply generate every possible W_{i,alpha} and compute
17      # the corresponding subspace trail
18      for i, alpha in product(range(1, k+1), range(1, 1<<n)):
19          w_i_alpha = vector(GF(2),
20                  [0]*(n*(k-i)) +
21                  to_n_bits(n, alpha) +
22                  [0]*(n*(i-1))
23              )
24          W_i_alpha = vs.subspace([w_i_alpha])
25
26          trails.append(compute_trail(f, W_i_alpha, dim))
27
28      return trails
```